AMENDMENTS TO THE SPECIFICATION

In paragraph [0028]: Fig.6 is an electronic system 600 having a host 602 capable of accessing a plurality of peripheral devices 604, 606, 608 on a single port 610 of an IDE bus or a SATA interface according to the present invention. The electronic system 600 includes the host 602, a controller 612, buffer memory 614, and at least a first peripheral device 604 and a second peripheral device 606. The first peripheral device 604 is an optical storage device and includes the optical pick-up 210, and the optical medium 212. The second peripheral device 606 is a flash card device and includes the flash card access device 316. Additionally, more peripheral devices 608 having other functions can be included. Each of the peripheral devices 604, 606, 608 is connected to the controller 612. The controller 612 is also electrically connected to the single port 610 of the IDE/SATA channel 616 and allows the host 602 to access each of the peripheral devices 604, 606, 608 through the single port 610. By modifying control codes and/or reserved vendor-specific bits in ATA Packet Interface (ATAPI) packets or registers in the IDE Task File that are sent ~~between~~ from the host 602 ~~and~~ to the controller 612, the host 602 is able to specify a target peripheral device. Using this structure, the three peripheral devices 604, 606, 608 can be implemented as a single product 601 and share the buffer memory 614 and other hardware (not shown). Additionally, the product 601 can have more than two peripheral devices and, because the controller 612 is only connected to a single port 610 of the IDE channel 616, other peripheral devices can still be attached to a second port 618 of the IDE channel 616 without loosing any of the functionally of the product 601 connected on the first port 610.

In paragraph [0030]: The host interface 700 electrically connects the controller 612 to the single port 610. Depending on configuration options, the host interface 700 can connect to the IDE~~/SATA~~ channel

616 as either a master or as a slave. The present invention is not limited to the selection of master or slave but rather that host interface 700 is electrically connected to one port 610 of the IDE channel 616 as only master or as only slave. In this way, the other port 618 (i.e. using the other master/slave setting) is free for use by another peripheral device connected to the IDE channel 616.

In paragraph [0031]: The internal memory 702 is used by the CPU as a temporary storage area and also includes program instructions corresponding to firmware code 714, which are executed by the CPU 704. It should be noted that the firmware code 714 could also be stored in external non-volatile memories (not shown) on or connected to the controller 612. The firmware code 714 contains instructions that allow the CPU 704 to read control codes and / or reserved vendor-specific bits in the ATAPI packets or registers in the IDE Task File that are sent ~~between~~ from the host 602 ~~and~~ to the controller 612. These control codes, reserved vendor-specific bits, and registers in the IDE Task File are referred to as a target device tag, which specifies a target peripheral device. The target peripheral device is one of the peripheral devices 604, 606, 608 connected to the controller 612. The CPU 704 accepts ATAPI commands from the host interface 700 and executes the commands according to the target device tag using the appropriate control unit. More specifically, if the target device tag specifies the optical storage device 604, the CPU 704 executes the ATAPI command using the optical storage control unit 708; if the target device tag specifies the flash card device 606, the CPU 704 executes the ATAPI command using the flash card control unit 710; and if the target device tag specifies another device 608, the CPU 704 executes the ATAPI command using the control unit for that device 712.

In paragraph [0036]: In one embodiment of the present invention, the optical storage device 604 is controlled by the default optical device

driver 808 using unmodified ATAPI commands. On boot-up or during initialization, the device driver 806 determines which other peripheral devices are connected to the controller 612 by reading a product model number from the controller 612. The device driver 806 then creates a plurality of virtual devices 812 corresponding to the other peripheral devices 606, 608 that are connected to the controller 612. The default optical device driver 808 provides a set of Application Program Interfaces (APIs) to user programs that need to access the optical storage device 604, and the default removable media driver 810 provides a set of APIs to the user space to allow user programs to access the other peripheral devices 606, 608 connected to the controller 612. The device driver 806 then modifies the target device tag for ATAPI packets that are sent to one of the other peripheral devices 606, 608 connected to the controller 612 other than the default device, which in this embodiment is the optical storage device 604. ~~Likewise, ATAPI packets that are received by the default optical device driver 808 from the controller 612 having modified target device tags are passed to the device driver 806.~~ In the controller 612, unmodified ATAPI packets are executed using the optical storage control unit 708 while modified ATAPI packets are passed to the appropriate control unit by the vendor specific functions code 802. Additionally, because some operations, such as high speed write operations to an optical medium, are very time critical, the (optional) scheduler 804 can be used to ensure that time critical ATAPI packets are passed to the controller 612 before non time critical ATAPI packets based on a priority ranking. The priority ranking can be a dynamic ranking that varies according to operations, such as write operations, or speed settings of the peripheral devices. For example, ATAPI packets being sent to a 12X speed digital versatile disc (DVD) writer are much more time critical than packets being sent to a flash card device or even a 2x speed DVD writer. In this way, peripheral devices having different timing requirements can be connected to the same controller 612 and reliably share the single port 610.

In paragraph [0041]: Step 904: Access the peripheral devices from the host using the single port of the predetermined interconnection means. By modifying a target device tag in packets that are sent from ~~between~~ the host ~~and~~ to the controller, a target peripheral device can be specified. Using this method, the M peripheral devices can share buffer memory and other hardware in or connected to the controller. Additionally, because the controller is only connected to a single port of the predetermined interconnection means, other peripheral devices can be attached to other ports without loosing any of the functionally of the M peripheral devices connected to the controller.

In the Abstract: An electronic system includes a host; a controller electrically coupled to the host through a single port of a predetermined bus, the single port for providing the host access to N devices; and M peripheral devices electrically coupled to the controller, M being greater than N. By modifying control codes, reserved vendor-specific bits in packets or registers in an IDE task file that are sent ~~between~~ from the host ~~and~~ to the controller, the host is able to specify a target peripheral device ~~and to determine which peripheral device sent each packet that is received by the host~~. In this way, the host can access the peripheral devices using the single port. When the peripheral devices include a first peripheral device and a second peripheral device, the controller can directly transfer data stored on the first peripheral to the second peripheral device without requiring the data to be buffered in the host.